Manual Técnico

Organización de Lenguajes y Compiladores 1

José Rafael Morente González | 201801237
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Contenido

Modelos

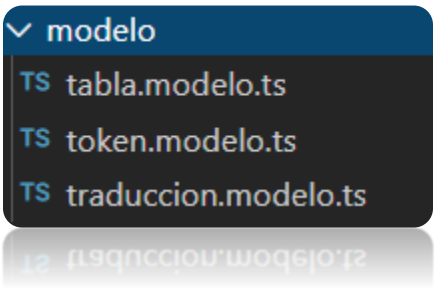


tabla.modelo.ts

Este modelo es utilizado por la tabla de simbolos para almacenar los valores de las variables.

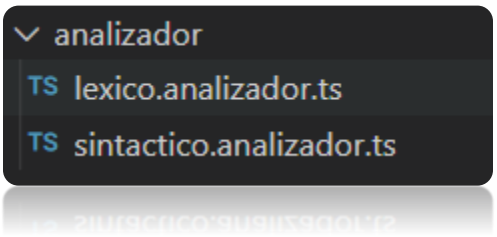
Token.modelo.cs

Este modelo es utilizado para administrar las propiedades del token en él analizador.

traduccion.modelo.cs

Este modelo es utilizado para administrar los tokens utilizados para traducir en él lenguaje.

Analizador



AnalizadorLexico.cs

El analizador léxico es la clase en la cual nosotros aceptamos cadenas de caracteres para nuestro lenguaje.

AnalizadorSintactico.cs

El analizador sintactico es donde nosotros verificamos el orden correcto de las palabras de nuestro lenguaje que queremos traducir para ello se utilizó un método llamado Match() el cual empareja los tokens correspondientes y su nivel de recuperación es a modo pánico hasta encontrar una palabra de la gramática el cual es punto y coma.

Análisis Léxico

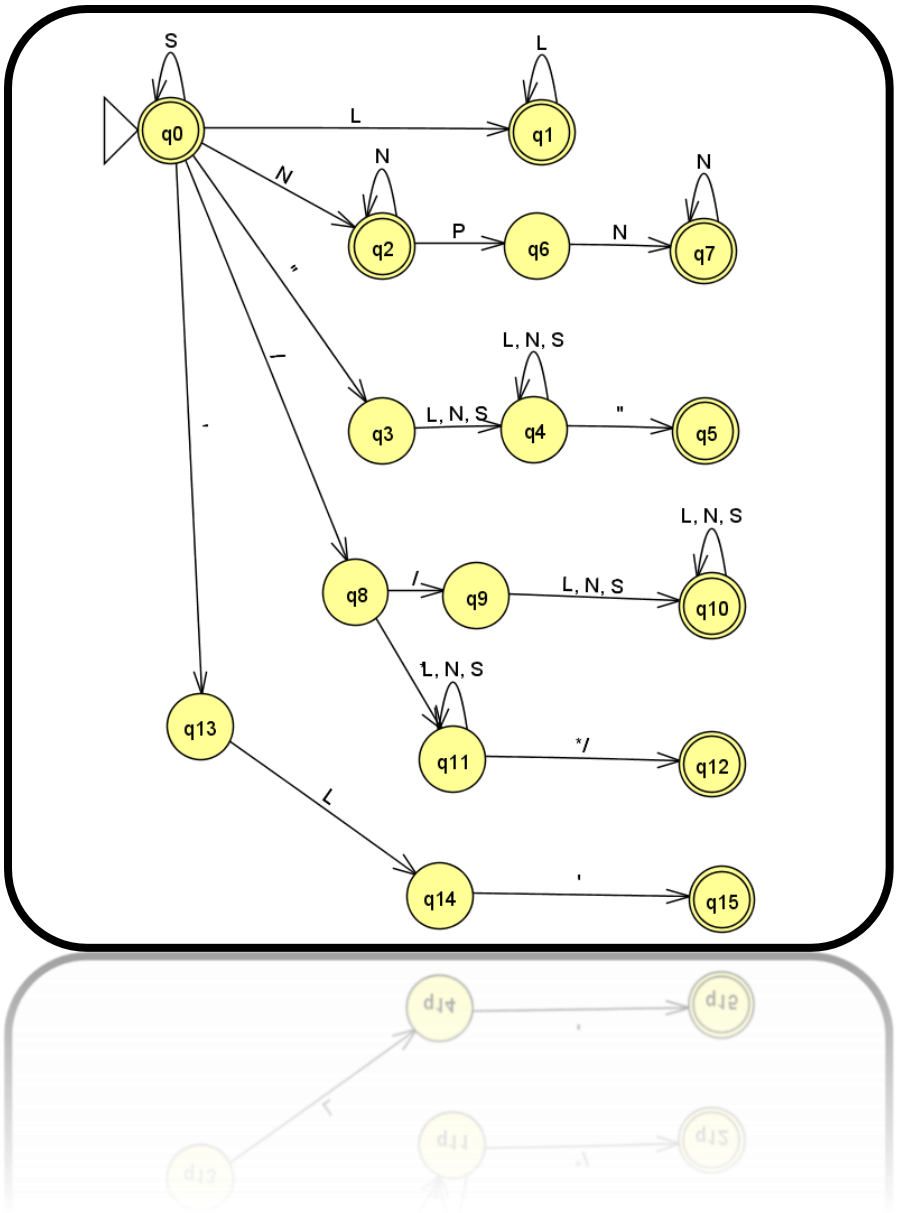
Conjuntos

$L = \{a, A, b, B, \dots, \dots, z, Z\}$

$S = \{Todos\ los\ Signos\}$

$\underline{N = \{0,1 \dots ,9\}}$

AFD



Análisis Sintáctico

<Inicio> ::= **<Clase>**

<Clase> ::= public class identificador llave_izquierda **<ListaDeclaracionGlobal>** llave_derecha

<ListaDeclaracionGlobal> ::= **<DeclaracionVariableGlobales>**

| **<MetodoPrincipal>**

| **<DeclaracionMetodo>**

| **E**

<ListaDeclaracion> ::= <DeclaracionVariable>

|<DeclaracionComentario>

|<DeclaracionIf>

|<DeclaracionFor>

|<DeclaracionWhile>

|<DeclaracionSwitch>

|<DeclaracionConsole>

|<DeclaracionSinTipo>

|<DeclaracionRetorno>

|<DeclaracionBreak>

<DeclaracionGlobal> ::= **<Declaracion> <OtraDeclaracionGlobal>**

<Declaracion> ::= **<MetodoPrincipal>**

| <TipoDatoGlobal>()

| <TipoDatoGlobal>,

|<TipoDatoGlobal>=12,

<TipoDatoGlobal>::= int identificador

<OtroMetodo> ::= **<Declaracion><OtraDeclaracionGlobal>**

<DeclaracionVariableGlobal> ::= <TipoDato><Funcion><ListaParametro>

<MetodoPrincipal> ::= void Main paréntesis_izquierda **<ParametroMain>** paréntesis_derecho llave_izquierda **<ListaDeclaracion>** llave_derecha

<TipoDato> ::= int

| doublé
| char
| bool

| string

<Inicio> ::= **<Clase>**

<Clase> ::= public class identificador llave_izquierda **<MetodoPrincipal>** **<DeclaracionMetodo>** llave_derecha

<MetodoPrincipal> ::= static void Main paréntesis_izquierda **<ParametroMain>** paréntesis_derecho llave_izquierda **<ListaDeclaracion>** llave_derecha

<ParametroMain> ::= string corchete_izquierdo corchete_derecho identificador

| E

<DeclaracionMetodo> ::= **<Metodo>** **<OtroMetodo>**

<Metodo> ::= void Identificador paréntesis_izquierda **<ParametrosMetodos>** paréntesis_derecho llave_izquierda **<ListaDeclaracion>** llave_derecha

| **<TipoDatoFuncion>** Identificador paréntesis_izquierda <ParametrosMetodos> paréntesis_derecho llave_izquierda **<ListaDeclaracion>** **<Retorno>** llave_derecha

<TipoDatoFuncion> ::= int

| doublé
| char
| bool

| string

<OtroMetodo> ::= **<Metodo><OtroMetodo>**

|E

<DeclaracionParametros> ::= **<TipoVariable>** **<ListaParametro>**

|E

<TipoVariable> ::= int

| doublé
| char
| bool

| string

<ListaParametro>::= identificador**<MasParametros>**

<MasParametros> ::= coma **< ListaParametro >**

|E

<Retorno> ::= return identificador;

<DeclaracionComentario> ::= **<Comentario><OtroComentario>**

<Comentario> ::= ComentarioLinea

| ComentarioMultilinea

|E

<OtroComentario> ::= **<Comentario><OtroComentario>**

|E

<ListaDeclaracion> ::= **<DeclaracionAsignacion>**

| <DeclaracionIf>

|

<DeclaracionAsignacion> ::= <Asignacion><OtraAsignacion>

<Asignacion> ::= **<TipoVariable> <ListaAsignacion><AsignacionVariable>** punto_coma

<TipoVariable> ::= int

| doublé
| char
| bool

| string

<ListaAsignacion>::= identificador**<MasElementos>**

<MasElementos> ::= coma **<ListaAsignacion>**

|E

<AsignacionVariable> ::= igual **<ValorVariable>**

|E

<ValorVariable> ::= digito

| cadena
| caracter
| identificador

| true

| false

<OtraAsignacion> ::= **<Asignacion><OtraAsignacion>**

|E

<DeclaracionIf> ::= if paréntesis_izquierdo **<CondicionIf>** paréntesis_derecho llave_izquierda **<Else> <ListaDeclaracion>**

<Else> ::= else **<TipoElse>**

|E

<TipoElse> ::= **<DeclaracionElseIf>**

| llave_izquierda <ListaDeclaracion> llave_derecha

<DeclaracionElseIf> ::= **<ElseIf><OtroElseIf>**

<OtroElseIf> ::= **<ElseIf><OtroElseIf>**

|E

<ElseIf> ::= if paréntesis_izquierdo <Condicion> paréntesis_derecho llave_izquierda **<ListaDeclaracion>** llave_derecha

| else llave_izquierda **<ListaDeclaracion>** llave_derecha

|E

<OtroElseIf> ::= **<ElseIf><OtroElseIf>**

|E

<Condicion> ::= <TipoVariable><OperacionRelacional><TipoVariable>

<TipoVariable> ::= identificador

| numero

| carácter

| cadena

<DeclaracionFor> ::= if paréntesis_izquierdo **<InicializacionFor>** punto_coma **<CondicionFor>** punto_coma **<IncrementoFor>** paréntesis_derecho llave_izquierda **<ListaDeclaracion>** llave_derecha **<ListaDeclaracion>**

<DeclaracionWhile> ::= while llave_izquierda **<ListaDeclaracion>** llave_derecha **<ListaDeclaracion>**

<DeclaracionSwitch> ::= if paréntesis_izquierdo identificador paréntesis_derecho llave_izquierda **<CuerpoSwitch>** llave_derecha **<ListaDeclaracion>**

<CuerpoSwitch> ::= **<Case><OtroCase><Default>**

<Case> ::= case **<TipoSwitch>** dos_puntos **<ListaDeclaracion> <Break>**

<TipoSwitch> ::= identificador

| numero

| carácter

| cadena

<OtroCase> :: **<Case><OtroCase>**

|E

<Break> ::= break punto_coma

|E

<Default> ::= default: **<ListaDeclaracion>** break punto_coma

|E

<OperacionRelacional> ::= menor

| menor igual

| mayor

| mayor igual

| igual igual

| exclamación igual

<DeclaracionDoWhile> ::= do llave_izquierda **<ListaDeclaracion>** llave_derecha while paréntesis_izquierdo **<Condicion>** paréntesis_derecho punto_coma

<InicioVariable> ::= <DeclaracionVariable>

<DeclaracionVariable> ::= <Tipo> <ListaId> <OpcAsignacion> punto_y_coma

<Tipo> ::= int

| string

| char

| float

| bool

<ListaId> ::= Identificador <ListaId2>

<ListaId2> ::= coma Identificador <ListaId2>

| ε

<OpcAsignacion> ::= Signo_igual <valor>

| ε

<valor> ::= int

| string

| char

| float

| bool

<Expresion> ::= <Termino> <ExpresionPrima>

<ExpresionPrima> ::= Simbolo_mas

| Simbolo_menos

| ε

<Termino> ::= <Factor> <TerminoPrima>

<TerminoPrima> ::= Simbolo_por

| Simbolo_div

| ε

<Factor> ::= Parentesis_derecho <Expresion> Parentesis_derecho

| Digito

| cadena

| Identificado

<DeclaracionSinTipo> ::= Identificador Simbolo_igual **<Expresion>** Simbolo_puntoYcoma **<ListaDeclaracion>**